Öğr.Gör. Mustafa GÖKMEN
gokmen@selcuk.edu.tr

Nesneye Yönelik Programlama

Java Variables

Variables (Değişkenler)

- *Fields (Java Attributes):* Bir değişken *sınıfın içerisinde, ancak bir metod veya kod bloğunun dışında* tanımlanmış ise bu değişken türüne *field* (alan ya da attribute) denir. Alanların kullanımı, tanımında static anahtar kelimesinin var olup olmadığına göre farklı şekillerde ele alınıp değerlendirilir.
  - ✓ *Class Variables (Static Fields) :* Eğer bir alan static anahtar kelimesi ile oluşturulmuşsa o alan sınıfın kendisine ait bir değişken olarak adlandırılır ve nesneler için fiziksel olarak tekrar üretilmez.

    Örnek: `public static int var = 1;`
  - ✓ *Instance Variables (Dynamic Fields) :* Eğer bir alan static anahtar kelimesi kullanılmadan oluşturulmuşsa o alan her nesnenin kendisi için ayrı ayrı oluşturulur ve sadece o nesneye ait olur.

    Örnek: `public int var = 1;`

- *Local Variables:* Metod veya kod bloklarının içerisinde tanımlanırlar. Method veya kod bloklarına girildiğinde oluşturulur ve çıkıldığında yok edilirler.

- *Constant Variables:* Değeri değiştirilemeyen (constant) bir değişken, *final* anahtar sözcüğü kullanılarak ve bir başlangıç değeri atanarak oluşturulur.

Fields
(Class Variable)

Class Variable, bir sınıftaki *static anahtar sözcüğü ile birlikte sınıfın içerisinde, ancak bir metod veya kod bloğunun dışında bir (field) alan olarak* oluşturulur. *Bu alan sınıfın kendisine aittir ve o sınıfa ait ilk nesne oluşturulduğunda fiziksel olarak var edilip heap bellekte saklanır, bir daha tekrar oluşturulmaz ve son nesne yok edilirken fiziksel olarak da bellekten atılır. Haliyle o sınıftan meydana gelen tüm nesneler tarafından ortak olarak kullanılan tek bir alan olacağından, her hangi bir nesne içerisinden değeri değiştirildiğinde o değer, diğer tüm nesneler içerisinde de geçerli olacaktır.*

Sınıf değişkenleri için şu kurallar geçerlidir:

- *Sınıf değişkeni için bellekte bir tek yer ayrılır ve sınıfa ait farklı nesnelerde farklı kopyaları olamaz.*
- *Sınıf değişkenlerine ayrılan bellek program bitene kadar etkindir ve ancak program bitince boşaltılır.*
- *Sınıf değişkenleri nesneye bağlı olmadıklarından, onlara nesne referansı olmadan erişilebilir.*
- *Sınıf değişkenine, başlangıç değeri verilmezse (değişken türüne göre) varsayılan değeri alır.*

```
public class YeniBirSinif {
    static int var = 1;
}
```

Her hangi bir nesneye ait olmadığından doğrudan ait olduğu sınıf adı ile "**.**" (nokta) kullanılarak ulaşılır.

```
System.out.println(YeniBirSinif.var);
```

ya da aynı sınıfta ise sadece değişken adı ile ulaşılır.

```
System.out.println(var);
```

Fields
(Instance Variable)

Instance Variable (Nesne Değişkeni), bir sınıftaki *static anahtar sözcüğü olmadan sınıfın içerisinde, ancak bir metod veya kod bloğunun dışında bir alan (field) olarak* oluşturulur. *Sınıfın tanımına göre üretilen her nesne için ayrı ayrı oluşturulur ve bir sınıftan yeni nesne oluşturulduğunda, sadece o sınıfa ait değerleri tutmak için kullanılır.*

Örnek değişkeni için şu kurallar geçerlidir:

- *Nesne yaratılırken her nesne değişkenine bellekte bir yer açılır.*
- *Nesne yok edilince, nesne değişkenine açılan yer de yok edilir.*
- *Nesne değişkeni, her nesne için ayrı ayrı oluşturulur ve nesnenin kendisine ait bir değeri saklar.*
- *Nesne değişkenine başlangıç değeri verilmezse (değişken türüne göre) varsayılan değeri alır.*

Nesne oluşturulmadan nesne değişkeni için bellekte bir yer ayrılmaz.

```
public class YeniBirSinif {
    int var; // Nesne değişkeni
}
```

`YeniBirSinif` tipinde `ybs` adlı bir nesne oluşturulduğunda `var` adlı nesne değişkenine ulaşmak için;

```
YeniBirSinif ybs = new YeniBirSinif();
ybs.var = 10;
```

ifadesi kullanılıdır.

Local Variables

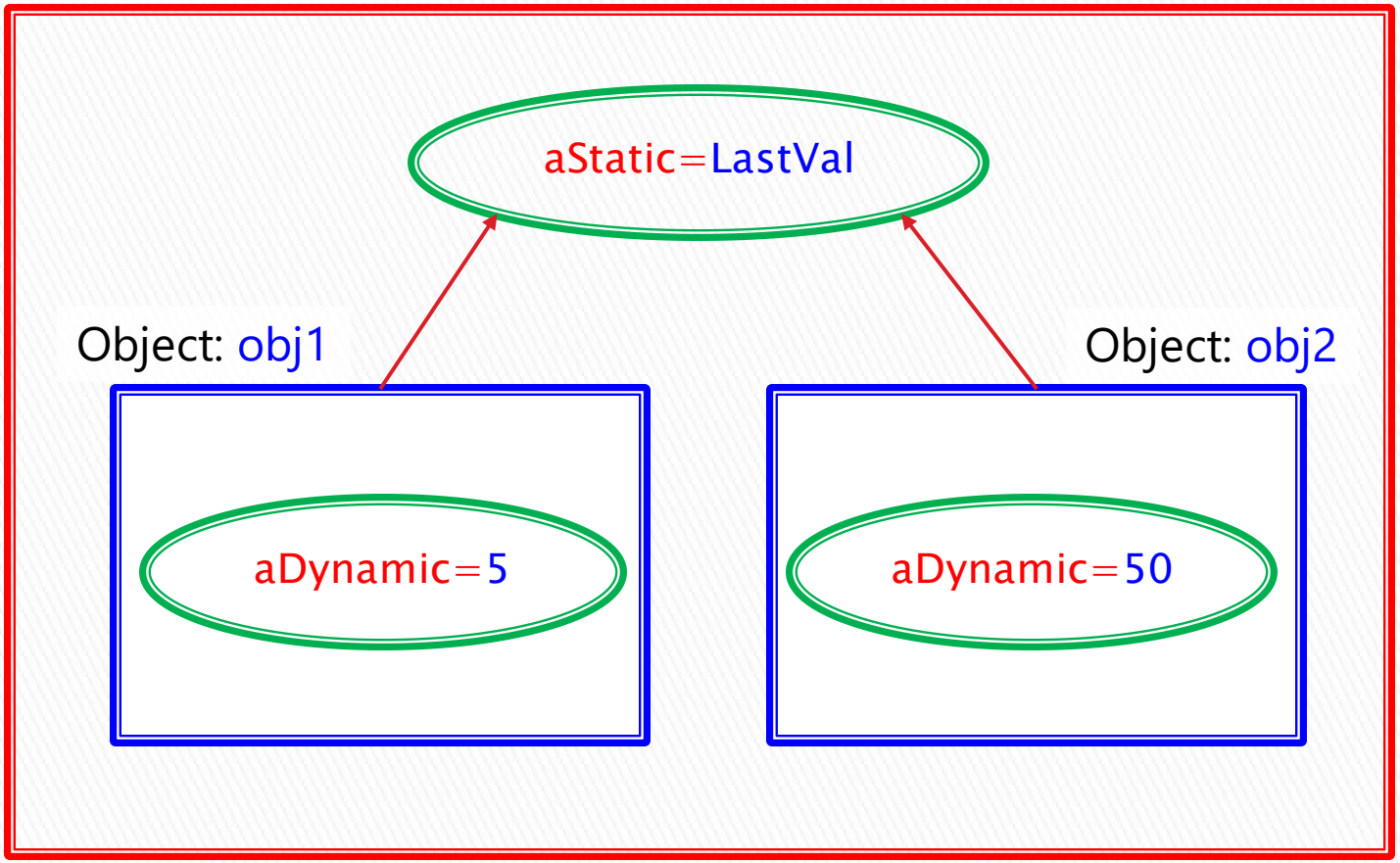
Local Variable, Metod veya kod bloklarının (scope) içerisinde tanımlanırlar. Metod veya kod bloklarına girildiğinde oluşturulur ve çıkıldığında yok edilirler. Local değişkenler bir metodun ya da bir kod blokunun dahili değişkenleridir. Her metod, sınıf içinde bir bloktur. Ayrıca {} içinde yazılı olan deyimler de metod içinde bir kod bloku oluşturur.

Yerel değişkenler için şu kurallar geçerlidir.

- *Metod veya blok çağrıldığında bellekte kendilerine birer yer açılır.*
- *Metodun veya blokun işi bitince, yerel değişkenler de bellekten silinir.*
- *Metodun veya blokun her çağrılışında, her yerel değişkene yeniden yer açılır.*
- *Yerel değişkenlere yalnızca ait oldukları metod veya blok içerisinden erişilebilir.*
- *Yerel değişkenler static olarak tanımlanamazlar.*

```
void hesapla(String kelime, int kdv) {
   static int lastVal = 1; // Geçersiz (static)
   int charLength = 10; // Geçerli
   { // yerel kod blokları
     int totalSize; // İlk değer atanmadığı için
     totalSize++; // bu satır hata verecektir.
   }
 }
```

Class: FieldTest



```java
public class FieldTest {
  static int aStatic;        // Class Variable
  int        aDynamic = 5;   // Instance Variable

  public static void main(String[] args) {
    FieldTest obj1  = new FieldTest();
    FieldTest obj2  = new FieldTest();
    char      aChar = 97;  // Local Variable

    int a = FieldTest.aStatic * obj1.aDynamic;
    System.out.println("1. Nesne için:");
    System.out.println(aChar + "Static = " + aStatic);
    System.out.println(aChar + "Dynamic = " + obj1.aDynamic);
    System.out.println(aChar + " = " + a);

    FieldTest.aStatic++;
    obj2.aDynamic = 50;

    a = FieldTest.aStatic * obj2.aDynamic;
    System.out.println("2. Nesne için:");
    System.out.println(aChar + "Static = " + aStatic);
    System.out.println(aChar + "Dynamic = " + obj2.aDynamic);
    System.out.println(aChar + " = " + a);
  }
}
```

```
1. Nesne için:
aStatic = 0
aDynamic = 5
a = 0

2. Nesne için:
aStatic = 1
aDynamic = 50
a = 50
```

Constants

Constant, değeri değiştirilemeyen (constant) bir değişken, *final* anahtar sözcüğü kullanılarak ve bir başlangıç değeri atanarak oluşturulur.

Sabitler, program çalışırken değer değiştiremezler. O nedenle, *final* özellikli değişkene, bildirimi yapılırken ilk değeri verilmelidir. Atanan bu ilk değer, program boyunca değişmeden kalır.

Diğer değişkenlerden ayırmak için, final değişkenlerin adlarını büyük harflerle yazmak bir gelenektir.

```
public class TestFinal {
  final double PI = 3.1415;

  public static void main(String[] args) {
    TestFinal fTest = new TestFinal();
    fTest.PI = 3.14; // Error : PI is a constant
    System.out.println(fTest.PI);
  }
}
```

Bir değişken final olarak tanımlanmış ise değeri değiştirilemez.

Final anahtar kelimesi

- final anahtar kelimesi, temelde değeri sabit olan (değiştirilemeyen) değişkenler, metodlar, parametreler ve sınıflar tanımlamada kullanılır.
- Bir alan/değişken final olarak tanımlanmış ise değeri değiştirilemez.
- Eğer bir sınıf final olarak tanımlanmışsa bu sınıftan yeni bir sınıf kalıtılamaz.
- Bir metod final olarak tanımlanmışsa bu metod override edilemez.
- Bir parametre final olarak tanımlanmışsa bu parametrenin değerini değiştirilemez.

Normal Metodlar

```
public class MethodNormal {
  void uyariYap() {
    System.out.println("Normal Method");
  }

  public static void main(String args[]) {
    MethodNormal mn = new MethodNormal();
    mn.uyariYap();
  }
}
```

MethodNormal.java uygulamamızda uyariYap() metodu statik değildir. Bu nedenle bu metodun çağrılabilmesi için *MethodNormal* sınıfına ait bir nesne (burada **mn** adı ile) oluşturulması gerekir.

Statik Metodlar

```
public class MethodStatic {
  public static void uyariYap() {
    System.out.println("Static Method");
  }

  public static void main(String args[]) {
    MethodStatic.uyariYap();
    // ya da aynı sınıfta ise
    uyariYap();
  }
}
```

Örnekteki tek fark `uyariYap()` metodunun static olarak değiştirilmesi değildir; çağrılma şekli de değiştirilmiştir. `uyariYap()` metodunu çağırabilmek için `MethodStatic` sınıfına ait bir nesne oluşturulması gerekmez. `Sinif_Adi.static_metod()` veya aynı sınıfa ait ise doğrudan `static_metod()` olarak da çağırılabilir.

`main()` metodu da aynı şekilde işler, fakat `main()` metodu tek başına çalışabilir uygulamalar için bir başlangıç noktasıdır.

static keyword kullanımı

- Bir metod veya bir değişken `static` tanımlaması ile oluşturulmuş ise, kendisine, bir nesne referansı olmadan erişilebilir. Çünkü `static` olarak tanımlanmış olan bir şey, nesneye değil, sınıfın kendisine aittir.

  Örnek: `MyClass.area();` ve `MyClass.age = 5;` gibi

- Metodlar içerisinde tanımlanan yerel değişkenler ve parametreler `static` olamaz.

- Aynı sınıf içerisinde, `static` metodlardan, static olmayan alanlara ve metodlara erişilemez.

- `static` metodlar override edilemezler. (Bu işlemleri daha sonraki derslerde göreceğiz.)

static keyword kullanımı

```
public class StaticField {
  static int a = 3;
  int        b;

  public static void main(String[] args) {
    StaticField sf = new StaticField();
    sf.b = a;
    sf.displayNum(sf.b);
  }
  public void displayNum(int param) {
    static int c = 2; // Error
    System.out.println("Sayi : " + param);
  }
}
```

b : Static değil

a : Static

Metodlarda `static` değişken olamaz.

Statik: Static olmayan metod ve alanlara, static metod içerisinden ulaşılamaz.

```
public class StaticTest {
    public static void main(String[] args) {
        int a = 3;
        int b = a;
        displayNum(b); // Error
    }

    public void displayNum(int pB) {
        System.out.println("sayi : " + pB);
    }
}
```